UNCLASSIFIED

AFIT/EN/TR96-01

Air Force Institute of Technology

Acquiring Consistent Knowledge

Eugene Santos Jr.   Darwyn O. Banks

January 3, 1996

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** January 1996	**3. REPORT TYPE AND DATES COVERED** Technical Report

4. TITLE AND SUBTITLE ACQUIRING CONSISTENT KNOWLEDGE	**5. FUNDING NUMBERS**
6. AUTHOR(S) Eugene Santos Jr. and Darwyn O. Banks	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/EN/TR96-01
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Distribution Unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

We develop a new methodology and tool for knowledge acquisition under uncertainty. A new knowledge representation called Bayesian Knowledge Bases provides a powerful key to our approach and is well-grounded in probability theory. In this paper, we demonstrate the ease and flexibility with which knowledge acquisition can be accomplished while ensuring the consistency of the knowledge base as data is both acquired and subsequently maintained. Furthermore, we handle issues such as temporal and default reasoning. We present the MACK tool and apply it to NASA's Post-Test Diagnostics System which locates anomalies aboard the Space Shuttles' Main Engines.

14. SUBJECT TERMS Knowledge Acquisition, Knowledge Engineering, Uncertainty, Knowledge Organization, Knowledge Consistency, Probabilistic Representation, Automated Acquisition, Knowledge Representation, Expert Systems, Diagnosis	**15. NUMBER OF PAGES** 43
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Acquiring Consistent Knowledge

Eugene Santos Jr.[1] and **Darwyn O. Banks**

January 3, 1996

Abstract

We develop a new methodology and tool for knowledge acquisition under uncertainty. A new knowledge representation called Bayesian Knowledge Bases provides a powerful key to our approach and is well-grounded in probability theory. In this paper, we demonstrate the ease and flexibility with which knowledge acquisition can be accomplished while ensuring the consistency of the knowledge base as data is both acquired and subsequently maintained. Furthermore, we handle issues such as temporal and default reasoning. We present the MACK tool and apply it to NASA's Post-Test Diagnostics System which locates anomalies aboard the Space Shuttles' Main Engines.

Keywords: Knowledge Acquisition, Knowledge Engineering, Uncertainty, Knowledge Organization, Knowledge Consistency, Probabilistic Representation, Automated Acquisition, Knowledge Representation, Expert Systems, Diagnosis

[1]This research was supported in part by AFOSR Project #940006.

1 Introduction

Knowledge engineering new domains remains a daunting task for both the expert and the engineer involved. The knowledge must be elicited from the expert and then converted into a form according to the internal knowledge representation of the target expert system. Furthermore, the knowledge itself must then be validated and verified to ensure the system's reliability. Figure 1.1 lays out the standard three phase knowledge acquisition process.

The ease with which we perform a given phase is intricately tied to each of the other phases. For example, the interview should ideally be as painless as possible avoiding problems such as redundant questioning, overly rigid response templates — requiring the expert to answer using an inflexible and often unrealistic format, etc. Only the most intuitive and flexible of knowledge organization should be required of the expert. Unfortunately, if the internal knowledge representation is significantly different from the organization of the interview, then the knowledge engineer is faced with the onus of properly encoding the information. This typically entails a radical re-structuring of the given information while simultaneously attempting to preserve its original content.

For the moment, let's assume that our knowledge representation is also relatively simple and mirrors the interview organization. While this simplifies the job for the knowledge engineer, we end up impacting our last phase. Clearly, there is a tradeoff between the amount of organization and flexibility inherent in our knowledge representation versus our ability to perform verification and validation over it. For example, the problem of consistency is especially sensistive. Without much organization, it is nearly impossible to detect when and where an incosistency has occurred.

There are many other factors that determine the success or failure of a knowledge acquisition tool. This includes many pragmatic concerns such as learnability of the tool, input requirements — user interface, and knowledge induction [12, 15]. All of the above factors serve towards building an ideal tool for assisting knowledge engineers. Probably the most important element shared by these factors is our choice of knowledge representation especially in light of our earlier scenario.

The major difficulty faced by all knowledge engineers is in dealing with uncertainty: uncertainty in the expert's themselves about their knowledge and uncertainty in the engineer trying to translate the knowledge. Although it seems that all knowledge can be encoded in logical "if-then" style rules — indeed, most agree that it is the simplest and most intuitive approach to organization [7], exceptions to the rule quickly explode the number of rules necessary. Unfortunately, this further leads to questions on completeness. Unless all the exceptions have been identified, the original rule being as general as it is will produce an incorrect result in the situations where the un-identified exceptions occurs.

A wide variety of models for uncertainty are available. However, we must be

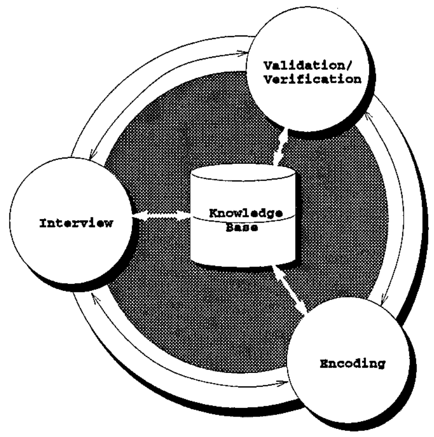

FIG. 1.1. Knowledge acquisition process.

careful in choosing an appropriate one for our task. Managing the uncertainty is also a task by itself and the most successful approaches ground themselves with strong semantics such as probability theory. This grounding helps to reduce general ad-hocness and reasoning anomalies which arise from weakly defined semantics. Furthermore, completeness and consistency are easier to guarantee. The trade-off, of course, again lies in the flexibility of the representation and most importantly in how intuitive it is for the expert and knowledge engineer to work with.

In this paper, we present a new knowledge acquisition tool called MACK. Its knowledge representation is semantically well-grounded yet remains extremely flexible and intuitive. Furthermore, MACK automatically guarantees that our knowledge-base is always consistent and assists the engineer in avoiding or correcting inconsistencies as knowledge is acquired. Our tool is also capable of dealing with temporal information as well as managing various forms of incompleteness. Finally, we demonstrate our system by applying it to the real world problem of Post-Test Diagnosis on the Space Shuttle Engines for NASA Lewis Research Center.

We begin in Section 2 by describing Bayesian Knowledge Bases as our model of knowledge. Section 3 then discusses how consistency is maintained both globally and incrementally for our model. Next, we present the real-world problem of Post-Test Diagnosis Systems for the Space Shuttle in Section 4 and then describe the MACK tool with which we developed for this domain in Section 5.

2 Managing Uncertainty

There is an inverse relationship between the efficiency of an automated reasoning algorithm and the flexibility of its associated knowledge representation. However, the former is secondary to the overall capabilities of the final product when weighed against the latter. Without an appropriate representation we cannot properly store our knowledge. Thus, system designers often give preference to flexible representations [11]. Nevertheless, real-world applications for intelligent or expert systems, such as those in space operations domains, require a balance of these two capabilities. As we shall see, Bayesian Knowledge Bases (abbrev. BKBs) [27] are just such a representation.

Inference mechanisms such as those in BKBs are noteworthy for their ability to represent uncertainty precisely because they marry the strong models of probability theory with an "if-then" rule structure similar to those rules which Shortliffe and Buchanan [7] found most natural for human experts. Reliance on the well-established laws of probability helps guarantee that BKBs are a sound and consistent representation of knowledge [27], and therefore, that the results they generate will not be inconsistent.

Probabilistic reasoning in intelligent systems exploits the fact that probability theory is and has been an accepted language both for the description of uncertainty and for making inferences from incomplete knowledge [18]. Using the semantics of probability theory, we designate random variables to represent the discrete objects or events in question. We then assign a joint probability distribution to each possible state of the world, *i.e.*, a specific value assignment for each random variable. This assignment allows us to reason over the set of probabilities [4].

Unfortunately, the size of this joint distribution can grow exponentially in the number of random variables making inferencing and knowledge acquisition computationally intractable [18, 25]. One way to address this complexity is by assuming many, if not all, of the random variables to be independent. However, while such independence assumptions significantly facilitate knowledge acquisition and ultimate resolution, if applied carelessly, they can oversimplify the problem statement such that the final answer loses considerable validity [18].

Bayesian approaches avoid oversimplification by couching their independence assumptions in terms of conditional dependencies. Let D, E and F be random variables. The conditional probability, $P(D|E,F)$, identifies the belief in D's truth given that E and F are both known to be true. If $P(D|E,F) = P(D|E)$, we say that random variables D and F are conditionally independent given E. In other words, once we know E is true, we can establish D's truth with or without any knowledge of F's [22]. We call D the *head* of $P(D|E,F)$ and $\{E,F\}$ the *tail*.

Bayesian philosophy holds that such conditional relationships — *e.g.*, $P(D|E)$ — are more in keeping with the way humans tend to organize knowledge [16, 34, 33]. Equation (1) below shows Bayes' Formula for computing these probabilities:

$$P(D|E) = \frac{P(E,D)}{P(E)} \tag{1}$$

We can view random variable D as a possible hypothesis (or set of hypotheses) held in advance and E as the actual evidence that was or will be generated. This formula shows how previous hypotheses should be modified in light of that new evidence.

Equation (2) manipulates Bayes' Formula to allow us to compute the joint distribution.

$$P(E,D) = P(D|E)P(E) \tag{2}$$

It generalizes to n variables as shown in Equation (3):[2]

$$\begin{aligned} P(D,E,F,G,H) &= P(D|E,F,G,H)P(E|F,G,H)* \\ &\quad P(G|F,H)P(H|F)P(F) \end{aligned} \tag{3}$$

More importantly to inferencing, the subsequent incorporation of the known independence conditions further reduces the amount of information we must actually store to be able to compute the required joint probability [22, 20, 18, 28, 31]. Thus, let us assume that random variable E is known to be conditionally independent from both F and H when the value of G is known, and D is likewise conditionally independent with knowledge of both E and G. Then, we can simplify Equation (3) further still:

$$P(D,E,F,G,H) = P(D|E,G)P(E|G)P(G|F,H)P(H|F)P(F) \tag{4}$$

These conditional dependencies can also be represented pictorially with a directed graph. Let A, B and C be random variables representing a traffic light, its associated vehicle detector and pedestrian signal, respectively. Figure 2.1 graphically depicts this network over these variables. Since the signal depends upon the light, we say that A is the parent of C. Similarly, B is the parent of A.

Now, assume we want for some reason to expand this set with a probability for the detector stating the likelihood of its being tripped during rush hour. Such an inclusion would introduce a cycle into our graph since the detector and traffic light cannot both depend upon the other. It becomes synonymous to the classic circular reasoning example: "If Smoke, then Fire" coupled with "If Fire, then Smoke."

As we can see, conditional independence at the random variable level is overly restrictive. We must proceed to an even finer level of distinction. This is

[2]Equation (3) shows one of the n! possible expansions. n is obviously equal to 5 in this example.

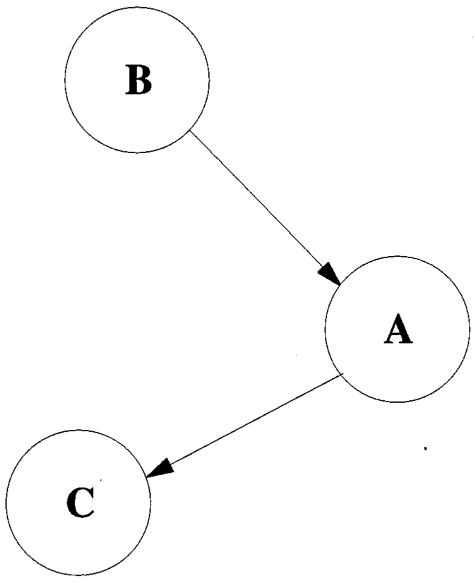

FIG. 2.1. Example graph with random variables as nodes.

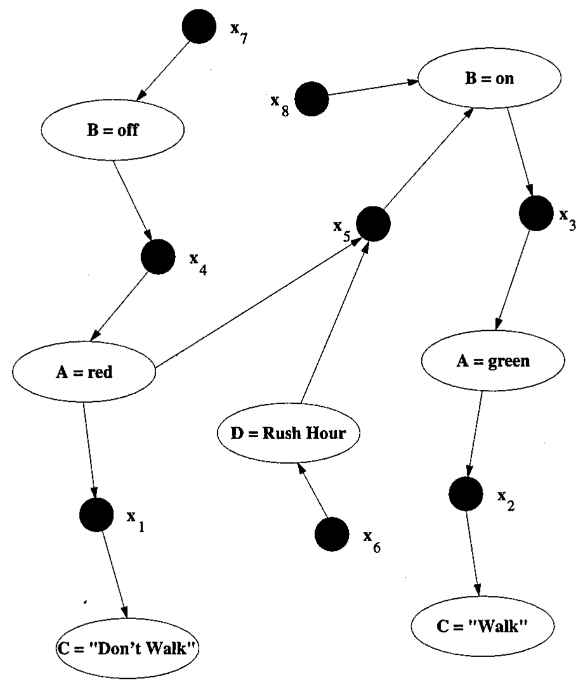

FIG. 2.2. Example Bayesian Knowledge Base.

the basis of the BKB representation. Assuming the same trio of random variables and the partial set of values below:

$$\mathrm{P}(\mathrm{C} = \text{“Don'tWalk''}|\mathrm{A} = \mathrm{red}) = x_1 \tag{5}$$
$$\mathrm{P}(\mathrm{C} = \text{“Walk''}|\mathrm{A} = \mathrm{green}) = x_2 \tag{6}$$
$$\mathrm{P}(\mathrm{A} = \mathrm{green}|\mathrm{B} = \mathrm{On}) = x_3 \tag{7}$$
$$\mathrm{P}(\mathrm{A} = \mathrm{red}|\mathrm{B} = \mathrm{Off}) = x_4 \tag{8}$$

We can quite legally add the new constraint:

$$\mathrm{P}(\mathrm{B} = \mathrm{On}|\mathrm{A} = \mathrm{red}, \mathrm{D} = \mathrm{rushhour}) = x_5 \tag{9}$$

without creating a directed cycle. Figure 2.2 shows the graphical representation of this example.

Notice how the graph of a BKB is a simple, bipartite and directed. It has two distinct types of nodes. The first, shown as lettered ovals, corresponds to individual random variable instantiations. However, now these nodes are particular not simply to a random variable, but to a specific instantiation thereof. Hence

Figure 2.1's single node for variable A, the traffic light, becomes two distinct instantiation nodes for "A = Red" and "A = Green" in Figure 2.2. These are called *instantiation nodes* or *I-nodes* for short.

The second type of node, depicted as a blackened circle, is called a *support node*. When drawn, these nodes, which represent the numeric probability value itself, have exactly one outbound arrow to the instantiation node representing the probability's head. Support nodes also provide a graphical terminus for zero or more inbound dependency or conditioning arrows from each of the parent instantiations in the probability's tail. Conversely, all instantiations nodes must have one or more inbound arrows to establish their probabilities and sets of supporting conditions, but need not contain any outbound arrows.

Reasoning algorithms for BKBs are built upon inference engines previously developed for weighted "AND/OR" directed acyclic graphs (abbrev. WAODAGs) [27, 29, 31, 28]. In these schema BKB support nodes correspond to WAODAG AND-nodes precisely because all parent instantiation nodes must be active or true before the support may be. Likewise, a BKB's instantiation nodes correspond roughly to the WAODAG's OR-node since any one of an instantiation node's support conditions is sufficient to activate it. The correspondence is inexact in that the BKB instantiation node actually represents an exclusive-or condition. Any support condition may substantiate an instantiation node, but that support node must be the **only** one active.

BKBs subsume many existing probabilistic models such as Bayesian networks [22]. They are more powerful from the fact that cyclicity can be represented in BKBs and that there is no requirement that all conditional probabilities must be specified unlike Bayesian networks. Furthermore, conditional independence conditions are much more explicit in the BKBs providing a clearer organization to the knowledge engineer.

3 Guaranteeing Consistency

As we have seen, BKBs allow cyclical construction, *i.e.*, we can have $\mathrm{P}(\mathrm{B} = \mathrm{b}_2 | \mathrm{A} = \mathrm{a}_2)$ and $\mathrm{P}(\mathrm{A} = \mathrm{a}_1 | \mathrm{B} = \mathrm{b}_2, \mathrm{C} = \mathrm{c}_1)$ in our database simultaneously. However, this additional constructive capacity changes the concept of consistency for a knowledge base in ways unique to BKBs.

BKB inference algorithms operate straightforwardly by computing a joint probability for a particular instantiation to all the random variables set — henceforth we will refer to such an assignment as a "state of the world." In an inconsistent knowledge base, there will be multiple ways to compute this value for any particular state of the world with no guarantee that the results will be equal as they must [27].

To illustrate, let X, Y and Z be boolean random variables. Clearly, there are eight possible states of the world. Given the following probabilities as the entire population of the BKB's database:

$$\mathrm{P}(\mathrm{X} = \text{true} | \mathrm{Y} = \text{false}) = .40 \tag{10}$$
$$\mathrm{P}(\mathrm{X} = \text{true} | \mathrm{Z} = \text{true}) = .80 \tag{11}$$
$$\mathrm{P}(\mathrm{Y} = \text{false} | \mathrm{X} = \text{true}) = .70 \tag{12}$$
$$\mathrm{P}(\mathrm{Y} = \text{false} | \mathrm{Z} = \text{true}) = .45 \tag{13}$$
$$\mathrm{P}(\mathrm{Z} = \text{true}) = .75 \tag{14}$$

The inference engine may compute $\mathrm{P}(\mathrm{X} = \text{true}, \mathrm{Y} = \text{false}, \mathrm{Z} = \text{true})$[3] by multiplying equations (10), (13) and (14) or by multiplying equations (12), (11) and (14). However, the joint probability on the first of these logical paths is .135, while along the second path it is .42, more than three times greater!

We eliminate this inconsistency by requiring any two tail-compatible[4] probabilities which share a head to have exactly equal probability values. In other words, we guarantee that the probabilistic value of equation (11) is the same as the value of equation (10) and that equation (13) is also equal to (12) as shown below.[5]

$$\mathrm{P}(\mathrm{X} = \text{true} | \mathrm{Y} = \text{false}) = .40 \tag{10a}$$
$$\mathrm{P}(\mathrm{X} = \text{true} | \mathrm{Z} = \text{true}) = \cancel{.80}.40 \tag{11a}$$
$$\mathrm{P}(\mathrm{Y} = \text{false} | \mathrm{X} = \text{true}) = .70 \tag{12a}$$
$$\mathrm{P}(\mathrm{Y} = \text{false} | \mathrm{Z} = \text{true}) = \cancel{.45}.70 \tag{13a}$$
$$\mathrm{P}(\mathrm{Z} = \text{true}) = .75 \tag{14a}$$

While this equality requirement clearly forbids inconsistencies, it does little either to explain or to assist the knowledge engineer in his or her efforts to build the system. The engineer's problem, then, is to determine the equating formula which will be used initially to create a viable knowledge base. Such formulae are countless—*e.g.*, minima, maxima, weighted or unweighted averages, any real function over the values, etc. Moreover, in the absence of other information, all can be equally valid. This makes an algorithm to construct BKBs all the more elusive.

[3]Note that there is insufficient data here to compute any other joint probability.

[4]We define probabilities to be *tail-incompatible* or *mutually exclusive* only if there exists a random variable in the tail of both which takes on a different value in each.

$$\mathrm{P}(\mathrm{Y} = \text{false} | \mathrm{Z} = \text{true}) = .45$$
$$\mathrm{P}(\mathrm{Y} = \text{false} | \mathrm{Z} = \text{false}) = .83$$

For example, these probability equations are tail-incompatible since both place conditions upon random variable Y's being false and the variable Z assumes a different value in their tails. Similarly conditioned probabilities with non-identical heads are considered *mutually exclusive*.

[5]In this case we have arbitrarily set the value of the second member of each pair equal to the first.

Before we can develop a knowledge acquisition methodology, we must be aware of those areas of a BKB with the potential to violate its construction constraints or to harbor inconsistent bits of knowledge. Once identified, we can ensure both the validity and consistency of a knowledge base by induction as it is being built. However, unlike inductive systems such as those predicated on the ID3 algorithm [23], BKBs have no requirement for the complete specifications of all attributes and values which make those systems less tenable for large data sets.

Following are descriptions of the eight construction constraints we determined, as well as the manner in which our implementation satisfies them. Taken together, they guarantee both the structure of the BKB and, more importantly, its probabilistic consistency. The fact that there are only eight underscores the flexibility of the BKB representation and its ability to obey the laws of probability theory while still being general enough directly to interface with the expert.

- **Constraint 1** - *All instantiation nodes must be the head of at least one support node.*
  This first constraint recognizes the fact that a BKB stores all its probabilistic information in its support nodes, rather than its instantiation nodes.
- **Constraint 2** - *The sources and sinks of a node must be well-defined.*
  Because we implement instantiation nodes and support nodes as separate but interrelated object classes, this constraint ensures that the instantiations referenced by any support node be well-defined.
- **Constraint 3** - *Each instantiation node represents a unique instantiation of a single random variable.*
  Here we simply guarantee that all instantiation nodes taken together form a set, not a family, *i.e.*, contain no duplicate instantiations.
- **Constraint 4** - *Any support nodes which share a head instantiation must be mutually exclusive.*
  Given any state of the world, all but one of an instantiation node's support conditions must conflict with that particular assignment of global values. In the parlance of logic, we could say that this constraint requires the truth or falsity of any instantiation node to be established via an exclusive-or condition among its attendant support nodes.

$$P(X = \text{true} | Y = \text{true}) = y_1 \tag{15}$$
$$P(X = \text{true} | Y = \text{true}, Z = \text{false}) = y_2 \tag{16}$$
$$P(X = \text{true} | Y = \text{false}, Z = \text{false}) = y_3 \tag{17}$$
$$P(X = \text{true} | Y = \text{false}, Z = \text{true}) = y_4 \tag{18}$$

For example, the equations above show a set of support conditions using boolean random variables: X, Y and Z. Clearly, each of these support conditions modifies the same head instantiation: setting X to "true."

However, probabilities (15) and (16) are not mutually exclusive, since the former, which does not depend upon random variable Z, will always be valid any time that the latter is. In this case when Y is true, either probability affords a valid inference path to substantiate X's truth, thus y_1 must equal y_2 for the database to be consistent.
Constraint 1 guarantees there will be one or more support nodes for each instantiation. This fourth constraint provides the necessary distinctions between those support nodes such that one, and only one, may be active.

- **Constraint 5** - *Given any inference chain, a support node's head must not occur in the tail of any of its successors in that chain.*
The fifth constraint closes the door on logical cycles. For example, equations (19) through (22) below form a loop in that all four can be simultaneously active,

$$P(A = a_1 | D = d_2) \quad (19)$$

$$P(D = d_2 | B = b_3, C = c_1) \quad (20)$$

$$P(B = b_3 | E = e_2) \quad (21)$$

$$P(E = e_2 | A = a_1, C = c_1) \quad (22)$$

i.e., no mutual exclusivities exist within the set, and equation (22) depends in part upon the head instantiation of a predecessor, in this case "$A = a_1$." Failure to preclude this cycle would allow the inference engine potentially to enter an infinite loop since equation (19) can clearly be re-investigated as a successor to (22).
As with many search problems, discovering these cycles can quickly become combinatorial. However, we can conduct this search *as the expert identifies support conditions* which ensures the consistency of the knowledge base. This also assists the expert in correcting inconsistencies by flagging them sooner, *i.e.*, upon introduction to the database, rather than later. We accomplish this search cheaply and efficiently using a depth-first algorithm which begins at the head of the new support node and branches throughout the BKB structure, as necessary.

- **Constraint 6** - *The instantiation nodes for a given random variable must not simultaneously appear in the head and tail of a support node.*
Because we are concerned only with one particular state of the world at a time, it is obviously unacceptable to have one instantiation of an item depend upon a different instantiation of that same item since both can never be true simultaneously. In fact, the probability shown in equation (23) and all others like it would always have to equal zero, per force.

$$P(Y = \text{true} | Y = \text{false}, Z = \text{true}) \quad (23)$$

Moreover, it is even less acceptable for the veracity of an instantiation to be dependent upon the instantiation itself as in equation 24.

$$P(Y = \text{true} | X = \text{false}, Y = \text{true}) \quad (24)$$

- **Constraint 7** - *At most one instantiation node for a given random variable can occur in the tail of a support node.*
  Following similar logic as Constraint 6, one item cannot be conditioned by multiple values of another precisely because the instantiations of the second item obviously contradict themselves. Thus, equation (25) is clearly invalid since the random variable X cannot both be true *and* false at the same time.
$$\mathrm{P(Y = true|X = false, Z = true, X = true)} \tag{25}$$
- **Constraint 8** - *Given any set of support nodes whose tails are mutually tail-compatible but whose heads each denote a distinct instantiation of a single random variable, the probabilities must sum to less than or equal to 1.*
  Because we are using a probabilistic reasoning scheme, we need this last constraint to disallow any simultaneously valid sets of probabilities for the same item from ever summing to values greater than 1. We use the fact that the head instantiations of the sets' elements share the same random variable to identify each set.
  In the example below, we have collected all support conditions for random variable A (regardless of instantiated value) which depend upon random variable B's first value.
$$\begin{aligned} \mathrm{P(A = a_1|B = b_1, C = c_1)} &= v \quad (26)\\ \mathrm{P(A = a_1|B = b_1, C = c_2, D = d_1)} &= w \quad (27)\\ \mathrm{P(A = a_2|B = b_1)} &= x \quad (28)\\ \mathrm{P(A = a_3|B = b_1, D = d_1)} &= y \quad (29)\\ \mathrm{P(A = a_3|B = b_1, D = d_2)} &= z \quad (30)\end{aligned}$$
  Equations (26) and (27) can never be active at the same time since they depend on different instantiations of C. Similarly, equation (30) is mutually exclusive both with equation (27) and equation (29) due to random variable D. Under this constraint, these dependencies on different states of the world divide this set of equations such that the following inequalities must all be true:
$$\begin{aligned} x + z &\leq 1 \quad (31)\\ v + x + z &\leq 1 \quad (32)\\ w + x + y &\leq 1 \quad (33)\\ v + x + y &\leq 1 \quad (34)\end{aligned}$$
  MACK automatically normalizes the values of any probabilities which violate this constraint simply by dividing each element of the set by the

total.[6] Notice that equation (28)'s probability, x, is a factor in all these subsets since it does not depend either on C or D and can thus be simultaneously true with any of the other four. We also note that in this example equation (32) effectively overrides (31) because if the former holds, then the latter must also *a fortiori.*[7]

A complete consistency check essentially involves verification of each of the eight aforementioned constraints. Obviously, such an approach may become computationally expensive, especially as the size of the network grows. However, we have implemented our BKB knowledge acquisition routine such that complete consistency checking is rarely necessary. Specifically, guarantors for Constraints 2, 3, 6 and 7 are built into the object creation routines. Thus, as the expert defines the items of the BKB, their associated values and dependencies, the software objects themselves prohibit duplicate instantiations and ensure the validity of all references between the instantiation node and support node classes. Constraints 1, 4, 5 and 8, then, are the only constraints explicitly tested during a review. In addition, our BKB implementation conducts such reviews incrementally. We check each new support node as it is entered into the database to ensure it introduces no new inconsistencies to the existing consistent BKB, *e.g.*, by engendering a logical cycle.

4 Post-Test Diagnosis

Development, maintenance and improvement of any large system, especially one with human lives at stake, usually involves extensive testing. NASA's Space Transportation System, or Space Shuttle, is no exception. Marshall Space Flight Center in Huntsville, Alabama routinely conducts ground tests and collects actual flight data on the shuttle's boosters to better assess the health, status and current capabilities of the reusable engines and their many components. Presently, these assessments involve large teams of engineers who review remote data received from hundreds of on-board sensors called PIDs. Officials then use these manual reviews to determine the fitness of the engine for another test or flight.

The Post-Test Diagnostic System is an on-going cooperative project to automate the Space Shuttle Main Engine (SSME) review process using intelligent systems. Its stated goals are:

- To aid in the detection and diagnosis of engine anomalies.
- To increase accuracy and repeatability of rocket engine data analysis.
- To reduce analysis time.

[6] The pre-normalized value is maintained in the system for use in subsequent normalizations over different sets of probabilities.

[7] v, w, x, y, z are all non-negative real-valued variables between 0 and 1.

When complete, its components will validate engine sensors, reliably extract salient features from telemetry data, and analyze SSME performance systems, combustion devices, turbomachinery and dynamics.

As of this writing, two different versions of one component—the High Pressure Oxidizer Turbopump (HPOTP)—have been built and validated by government contractors under the auspices of researchers at NASA Lewis Research Center in Cleveland, Ohio.[8] These systems provided us the opportunity to test MACK's applicability to a real-world domain and a set of known parameters against which to corroborate the utility of MACK-acquired knowledge for BKB reasoning.

The HPOTP is an engine component designed initially to raise, then to maintain the pressure of the liquid oxygen flowing into the engine at the varying levels of thrust during the shuttle's flight profile [1]. Using a turbine powered by the oxidizer preburner's hydrogen-rich hot gas, this centrifugal pump manages the flow of liquid oxygen into the engine's main and preburner injectors. Beside the pumps and turbines, the HPOTP's third major group of subcomponents contains the extensive shaft seals which separate pumps, turbines and the fluids they regulate [1].

Being an automated tool, MACK is designed to be operated directly by the domain expert. In fact, it is a key component of the PESKI Knowledge Organization and Validation subsystem currently under development for Air Force Office of Scientific Research (AFOSR). (See Appendix C on PESKI.) As a result, we, the knowledge engineers, simulated the expert's involvement. We note, however, that much of the previous knowledge engineering accomplished for HPOTP has involved collating and sorting the information gathered in numerous interviews with the Alabamian crew of rocket scientists.[9] Appendix A correlates selected text from knowledge acquisition interviews with anomaly definitions from the second version of HPOTP and with conditional probabilities in the HPOTP BKB. These correlations show that it is, in fact, plausible partially or completely to remove the middleman and allow the expert to be his/her own knowledge engineer—*i.e.*, if the expert is so inclined, s/he can with minimal instruction create a BKB from scratch.

5 The MACK Tool

Having described both BKBs and the HPOTP application, we now explore some of the processes by which MACK acquires knowledge. The PESKI architecture within which MACK resides assumes the knowledge engineer to be optional (see

[8]The first was developed by Science Applications International Corporation, San Diego, California [2]. The second by personnel from Aerojet Propulsion Division, Sacramento, California [6].

[9]NASA Lewis researchers have been conducting interviews with Marshall Space Flight Center engineers since Spring 1992.

Figure C.1). As a result, the MACK tool, like those discussed by Sandahl [26], can potentially be the primary interface with the expert. This role places a premium on user-friendliness as much as adherence to BKB constructs and probabilistic formalisms.

MACK is a menu-driven system. These menus allow us to handle the simpler BKB constraints — Constraints #2, 3, 6 & 7 (see Section 3)—by simple manipulation of the menu options presented to the user. Other illegal choices simply trap program control until a valid selection is entered. The examples excerpted below are taken from the HPOTP application.[10] It includes data entry of the conditions governing the shift anomaly noted via sensor, PID 990, here called "Anomaly 990 Shift." This anomaly depends upon the sensor's peak and equilibrium values which are represented by the random variables, "PID 990 Peak" and "PID 990 Equilibrium," respectively. Here the expert is creating the first support condition for the instantiation of the random variable "Anomaly 990 Shift" to value "Found."[11]

```
At present, [Anomaly 990 Shift]'s being [Found] depends upon the fol-
lowing sets of conditions:

    No support conditions!

Enter 0 to add new support conditions for [Anomaly 990 Shift]'s being
[Found].

Otherwise, enter 1 to quit
 0
```

Recall that Constraint 2 requires the instantiations and supports connected in a BKB to be well-defined. Thus the system when creating a support condition only presents a choice among the previously instantiated random variables. MACK additionally restricts the options to those variables which can actually be used in the nascent support condition. We can see this in the absence of Anomaly 990 Shift itself from the subsequent menu shown below which is in keeping with Constraint 6.

```
[Anomaly 990 Shift]'s being [Found] can depend upon which of the
following components:

    2 – PID 990 Equilibrium
    3 – PID 990 Peak
    0 – None of the Above Components

Choice: 2
```

[10]See [5] for complete transcripts.

[11]Anomaly variables are basically boolean: "Found" or "Not Found."

```
    1 – Nominal
    2 – Out of Family
    0 – None of the Above; Abort

Choice: 2
```

Having already selected a value of PID 990 Equilibrium, the expert is now queried for continuance. In this abbreviated example we see that the only remaining random variable option available to the expert is PID 990 Peak. Anomaly 990 Shift has been previously removed under Constraint 6 and PID 990 Equilibrium, as a new addition to the condition, is now illegal in accordance with Constraint 7.

```
Presently, this condition holds that [Anomaly 990 Shift]'s being [Found]
can depend upon the following:

    PID 990 Equilibrium = Out of Family

Do you wish to extend this condition? Y / N
y

[Anomaly 990 Shift]'s being [Found] can depend upon which of the
following components:

    3 – PID 990 Peak

Choice: 3

    1 – Nominal
    2 – Out of Family – High
    3 – Out of Family – Low
    0 – None of the Above; Abort

Choice: 2

Presently, this condition holds that [Anomaly 990 Shift]'s being [Found]
can depend upon the following:

    PID 990 Equilibrium = Out of Family
    PID 990 Peak = Out of Family – High

Do you wish to extend this condition? Y / N
n

Please complete the sentence below from the following list of choices:

    0 – inconceivable
    1 – not likely
    2 – possible
```

3 – probable
4 – almost certain

It is ______________ that the [Anomaly 990 Shift] is [Found] depending upon ...

PID 990 Equilibrium = Out of Family
PID 990 Peak = Out of Family – High

Choice: 3

MACK then presents the expert with a menu of choices from which it will internally derive the support condition's probability. Since a BKB's probabilistic nature is masked from the expert, we use only the qualitative and linguistic terms shown below with their current value ranges.[12]

inconceivable:	0.00 - 0.10
not likely:	0.10 - 0.35
possible:	0.35 - 0.65
probable:	0.65 - 0.90
almost certain:	0.90 - 1.00

It is important to note here that during knowledge acquisition for a BKB, the actual numeric value assigned to any given probabilities is not significant. Refinement of these values through belief revision and belief updating is the province of the reasoning and explanation facilities in PESKI. The values associated with each node only attain meaning after the inference engine reasons over them during belief updating. It should be obvious, however, the inference engine's propagation of probabilities must begin somewhere. In his discussion of the validity of such values to probabilistic reasoning schemes, Pearl [22] writes:

> [p. 148, The] conditional probabilities characterizing the links in the network do not seem to impose definitive constraints on the probabilities that can be assigned to the nodes. ... The result is that any arbitrary assignment of beliefs to the propositions a and b can be consistent with the value of $P(a|b)$ that was initially assigned to the link connecting them ....

Thus, the decision to use a random number generator in the initial stages of database development neither adds to nor detracts from the BKB. More germane to the topic at hand, it certainly has no impact upon the consistency of the data's logical organization within that BKB.

These menu restrictions only account for the simple constraints. The more involved BKB formalisms are found in a separate consistency checking routine. MACK initiates this larger routine itself after any change to the set of support conditions, removal of an instantiation or upon receipt of up to five new, unsupported instantiations.

[12]Other research in this area has shown the commonality of adjective connotations. Current work seeks to refine these further [13, 10, 4, 9, 14, 35].

```
Welcome to M.A.C.K. — The BKB Module for the
       Acquisition of Consistent Knowledge!!

    0 – Generate new BKB
    1 – Edit existing BKB
    2 – Display current BKB
    3 – Load BKB from file
    4 – Save BKB to file
    5 – Check Knowledge Base Consistency
    6 – Run BKB Belief Revision Program
    7 – Delete the current BKB
    8 – Exit BKB program
```

This consistency checking routine sampled below covers the four remaining BKB constraints and, as a courtesy, also notifies the user of any conditions with zero probability. Initially, we see below that the knowledge base has failed Constraint 1 since the system has no condition defining a probability value for the absence of Anomaly 990 Shift. In these cases, MACK prompts the expert appropriately. Were the expert to answer any of these negatively, the consistency routine aborts there and returns the expert to the edit menu.

```
This BKB is currently inconsistent.

Is it correct that

    [Anomaly 990 Shift] being [Not Found]

does not depend on anything else? Y/N
y

Please complete the sentence below from the following list of choices:

    0 – inconceivable
    1 – not likely
    2 – possible
    3 – probable
    4 – almost certain

It is ______________ that the [Anomaly 990 Shift] is [Not Found]
depending upon ...

    Nothing!

Choice: 3

This BKB is currently inconsistent.
```

```
Is it correct that

     [PID 990 Equilibrium] being [Nominal]

does not depend on anything else? Y/N
 n

Please edit the conditions for

     [PID 990 Equilibrium] being [Nominal]

accordingly.

Growing the graph for your BKB.
Instantiations:

     0 – Add new instantiation
     1 – Delete instantiation

Support Conditions:

     2 – Add new support condition
     3 – Edit existing support condition
     4 – Delete support condition

     5 – Return to main menu
```

Constraint 4 is an important one which identifies support conditions that are not mutually exclusive. With insufficient information to make any automatic resolution assumptions here, MACK again queries the expert.

```
ERROR: Support conditions below are not mutually exclusive.

At present, [Anomaly 990 Shift]'s being [Found] depends upon the fol-
lowing sets of conditions:

     Support Node #1:

          PID 990 Equilibrium = Out of Family
          PID 990 Peak = Out of Family – High

     Support Node #2:

          PID 990 Peak = Out of Family – Low
          PID 990 Equilibrium = Nominal

     Support Node #3:

          PID 990 Equilibrium = Nominal
```

```
This BKB is currently inconsistent.

The following pair of conditions for [Anomaly 990 Shift] being [Found]
are not mutually exclusive.

    First Set:

        PID 990 Peak = Out of Family – Low
        PID 990 Equilibrium = Nominal

    Second Set:

        PID 990 Equilibrium = Nominal

Does [Anomaly 990 Shift]'s being [Found] really depend

    upon both sets of conditions? [Enter 0]

        or

    upon each set separately? [Enter 1]

Choice: 1

Which of these conditions may we add to eliminate the overlap?

    1 – [PID 990 Peak] can be [Nominal]
    2 – [PID 990 Peak] can be [Out of Family – High]
    0 – None of the Above

Choice: 2
```

The expert is given the option either of merging the two conditions into one or of distinguishing them in some way. While the first option is straightforward,[13] the second could conceivably draw upon any component in the BKB except the one in question, *i.e.*, the head of the two support conditions. To assist the expert in this area, MACK makes an initial simplifying assumption that excludes all random variables that are not already present. Since the only way to establish mutual exclusion is for both of the support conditions to contain in their tails a different instantiation of one or more variables (see Section 3). The basis of the assumption is that at least one of the current random variables can be expanded to meet this requirement, thus allowing the tool automatically to select and present options as it does in other areas. These options will be all the values of the existing variables which are not already represented. MACK can easily determine which of the two support nodes should obtain the adjustment

[13]This merger cannot be illegal, *i.e.*, violate either Constraint 6 or 7. If the support conditions in question reference instantiations which will conflict when merged, then they are mutually exclusive and, therefore, not inconsistent.

since, of course, Constraint 7 which proscribes against multiple values remains in effect.

Verifications of Constraint 8, shown below, occur somewhat innocuously. Since the expert is not aware of the actual probabilistic values anyway, MACK can simply normalize the pertinent sums[14] and reports any adjustments of the support conditions' qualitative variables — *e.g.*, inconceivable, not likely, possible, probable, or almost certain — to the expert. These normalizations always use the original probabilistic range the expert assigned in order to avoid a new, high-value addition from overwhelming predecessors whose values may have already been reduced.

```
This BKB is inconsistent.

Currently, support ranges overlap. Adjusting ranges for consistency ...
Conditions were:

    It is probable that the [Anomaly 990 Shift] is [Found] de-
    pending upon ...

        PID 990 Equilibrium = Out of Family
        PID 990 Peak = Out of Family - High

    It is probable that the [Anomaly 990 Shift] is [Not Found]
    depending upon ...

        Nothing!

New conditions are:

    It is possible that the [Anomaly 990 Shift] is [Found] depend-
    ing upon ...

        PID 990 Equilibrium = Out of Family
        PID 990 Peak = Out of Family - High

    It is possible that the [Anomaly 990 Shift] is [Not Found]
    depending upon ...

        Nothing!
```

The HPOTP application turned out to be a rather flat knowledge base. By that we mean that the sensor readings which represent the bulk of the random variables are unconditioned, and most anomaly determinations depend directly on the sensors rather than on some intermediate calculations. As a result, the application did not violate Constraint 5 which searches for logical cycles in the knowledge base.

[14]It is worth noting that although the normalization itself may be a trivial operation, the determination of the support node sets which are eligible to be normalized is not.

5.1 Temporal Information

Many real-world domains require the capability to model knowledge that changes over time. This requirement is even more pronounced in the PTDS domain which NASA eventually hopes to operate in real-time [6]. For a knowledge acquisition tool such as MACK, this introduces new developmental difficulties. While we want the interface with the expert to remain responsive and user-friendly, we must of course maintain the formalities of the system's probabilistic requirements.

Temporal aspects of HPOTP are best illustrated by the fact that the sensor readings are anything but static. During any given test, a sensor may take on multiple values, *e.g.*, Nominal, Erratic, Spiked, etc. Clearly, this violates the BKB's requirement that variable instantiations be unique and mutually exclusive. However, by partitioning the test period into time slices we can accommodate these changes.

For purposes of MACK's knowledge acquisition, the tool queries the expert for the number of changes in a particular temporal variable to expect at run-time. It then uses the largest of these when creating the BKB. Appendix B demonstrates our method of parsing these changes into the timeline. Presently, MACK arbitrarily limits a support node's relative time dependencies to the current time period and either adjacent period—*i.e.*, the ones immediately before or after T_i[15] — however, expansion of this range to $\pm n$ intervals, if necessary, is very straightforward.

The MACK interface's approach to the time-dependencies within a domain follow the similar protocols to those outlined for the constraints above. The distinctions reside in the way the tool queries the expert for the temporal dependency of each new random variable and the special menu options which accommodate those variables so identified.

```
Please enter the name of the new component:

    Item Name: PID 990 Peak

[PID 990 Peak] is a new component.
New INODE created: PID 990 Peak = No value given!!

REMINDER:

    BKB Variables are persistent and mutually exclusive. In other
    words, they take on 1, and only 1, of their possible values.
    Obviously, this will not accommodate variables that change
    over time.

Can the value of [PID 990 Peak] vary with time? Y / N
```

[15]In keeping with the system's intended flexibility, these qualitative temporal choices are "naturally linguistic" based on Allen's temporal model [30, 3].

```
y

How many times might [PID 990 Peak] change? 5
```

When these temporal components are encountered during construction, MACK again queries the expert for the appropriate relative time dependencies of the support condition. We have now included temporal conditions into our previous example shown below. The "< UNDEFINED >" linguistic variable is a placeholder pending the expert's assignment of a legitimate probabilistic range after entering all the tail values.

```
At present, [Anomaly 990 Shift]'s being [Found] depends upon the following sets of conditions:

    Support Node #1:

        PID 990 Equilibrium = Out of Family
        PID 990 Peak = Out of Family – High

Enter 0 to add new support conditions for

    [Anomaly 990 Shift]'s being [Found]

Otherwise, enter 1 to quit
0

[Anomaly 990 Shift]'s being [Found] can depend upon which of the following components:

    2 – PID 990 Equilibrium
    3 – PID 990 Peak
    0 – None of the Above Components

Choice: 3

    1 – Nominal
    2 – Out of Family – High
    3 – Out of Family – Low
    0 – None of the Above; Abort

Choice: 3

It is < UNDEFINED > that the [Anomaly 990 Shift] @ T_i is [Found] depending upon ...

    PID 990 Peak @ T_i = Out of Family – Low

New addition, [PID 990 Peak], is time-dependent. We should read its value, [Out of Family – Low], from which time interval?
```

```
    -1  –  Time period immediately preceding Anomaly 990 Shift = Found
     0  –  The same time period as Anomaly 990 Shift = Found
     1  –  Time period immediately after Anomaly 990 Shift = Found

0

Presently, this condition holds that [Anomaly 990 Shift]'s being [Found]
can depend upon the following:

    PID 990 Peak = Out of Family – Low

Do you wish to extend this condition? Y / N
y

[Anomaly 990 Shift]'s being [Found] can depend upon which of the
following components:

    2 – PID 990 Equilibrium

Choice: 2

    1 – Nominal
    2 – Out of Family
    0 – None of the Above; Abort

Choice: 1

It is < UNDEFINED > that the [Anomaly 990 Shift] @ T_i is [Found]
depending upon ...

    PID 990 Peak @ T_i = Out of Family – Low
    PID 990 Equilibrium @ T_i = Nominal

New addition, [PID 990 Equilibrium], is time-dependent. We should
read its value, [Nominal], from which time interval?

    -1  –  Time period immediately preceding Anomaly 990 Shift = Found
     0  –  The same time period as Anomaly 990 Shift = Found
     1  –  Time period immediately after Anomaly 990 Shift = Found

-1

Presently, this condition holds that [Anomaly 990 Shift]'s being [Found]
can depend upon the following:

    PID 990 Peak @ T_i = Out of Family – Low
    PID 990 Equilibrium @ T_i-1 = Nominal
```

```
Do you wish to extend this condition? Y / N
n

Please complete the sentence below from the following list of choices:

    0 – inconceivable
    1 – not likely
    2 – possible
    3 – probable
    4 – almost certain

It is ____________ that the [Anomaly 990 Shift] @ T_i is [Found]
depending upon ...

    PID 990 Peak @ T_i = Out of Family – Low
    PID 990 Equilibrium @ T_i-1 = Nominal

Choice: 2
```

5.2 Default Information

Default values for a BKB's random variables become important in cases where the system has incomplete information in the knowledge base with which to reason. These can include instances where the expert excludes pertinent data points, intentionally or otherwise, or instances wherein his/her implicit assumptions about the domain are not explicitly entered into the program.[16]

Currently, the PESKI architecture handles these situations via the Explanation and Interpretation Facility (See Figure C.1) which will present the system's output to the human expert to ensure the correctness both of the inference engine's results and the logical choices made to arrive at those results. If the value of a random variable is assumed in order to reach a solution, that variable must be flagged to the user. In the absence of essential information, these explanations will very likely contain assumptions of which the expert may approve or disapprove. In either case they will highlight the system's lack of information prompting the expert for its inclusion.

We accomplish this by modelling incomplete information about a random variable with probabilistic sums that are strictly less than 1.[17] There are two very distinct approaches, then, by which to account for the deficit: we can define

[16]The former type of incomplete information represents necessary knowledge which the expert neglected to incorporate, while the latter is information which by virtue of expertise s/he considers self-evident. In either case if the missing data causes the inference engine to assume a value, the system will highlight the deficiency to the expert.

[17]As we have seen in Section 3, the sum of probabilities for tail-compatible instantiations must not exceed 1. However, BKBs have no requirement for the support conditions' sums always to equal 1 either.

default values as *supersets* of the given support conditions or as the *negation* of those same rules.

The superset approach absorbs the probability values of existing rules, if any, for the default instantiation into a larger set which includes them and all the undefined probability values, *i.e.*, that fraction which brings the overall sum up to 1. In this case, we assume the default to be the most common occurrence and the rules to be identified exceptions thereto. "Default as negation" instead of adjusting the support conditions entered by the expert, simply creates an assumption node with a probability value which completely or partially complements the others, *i.e.*, its addition to the set maintains a sum less than or equal to 1.[18] Unlike the superset default, this assumption node can be activated if and only if all other support conditions fail. Now, the default—*i.e.*, the assumption node—has become the exception and the given rules constitute the common cases.

For an example let's return to the simple traffic light. Suppose each of the signal's three states—red (the default state), yellow and green—each has a probability of 0.3 with the remaining 0.1 unassigned. Superset default logic would absorb the uncounted 0.1 into the default's value such that red's value increases to 0.4. However, it masks the original, defined condition for the light's being red which may contain important information about the "red" state. The traffic light is simply assumed to be red unless the conditions for yellow or green are satisfied.

Negation, on the other hand, does not change any of the known data. Instead, it places part of the 0.1 value into an Assumption node. Then, if and only if no existing support condition for the traffic signal can be satisfied **and** the signal's value is pertinent to the solution being generated will the inference engine assume a value for the light. Notice that this assumed value need not be red. It will instead be the value that best supports the current solution, red or otherwise.

In a probabilistic system the primary concern for default reasoning is the continued adherence to the laws of probability. Clearly, the superset approach does this, however, in the process it can also give too much credence to its default value, and possibly remove seminal knowledge established by the expert. The resultant system runs the risks of choosing the default value more often than is warranted solely because of its artificially higher probability. Likewise, "default as negation" also maintains probabilistic validity, but without the artificial inflation of values as the assumption node is kept separate from all others. Its value may increase, but only with the express approval of the expert. Nevertheless, this latter approach can also become preferential to its previous

[18]The alternative to this assumption node is to completely specify all the possible instantiations that have not yet been enumerated and then assign explicit values to each. Needless to say, this solution paves the way for a combinatorial explosion in the required number of such instantiations, many of which do not serve to enhance the knowledge base since we can assume the expert would have included them if they did.

assumptions in those cases where the defined probabilities all have small values.

MACK supports the negation approach. Initially, we defend against over-reliance upon the default by guaranteeing the probability of the assumption node to be one or more orders of magnitude lower than any legitimate assignment.[19] However, during feedback the expert may give her/his permission to incorporate an assumption as an actual default support condition in its own right. In this case, we increase its value according to the probability range the expert assigns and continue to reason normally.[20] Constraint 8 (see Section 3) will then ensure the probabilities' sum is less than 1. Also, since the head of this default support condition can duplicate an existing instantiation but will not have a tail of its own, we must also bypass Constraint 4's requirement for uniqueness. Specifically, a default support node is, by definition, mutually exclusive with the set of all others, thus it must, of course, be tail-incompatible with any individual member of that set.

In diagnostic domains such as HPOTP, we are further protected from over-reliance on a default since the sensor readings are given for any test. Deduction on these readings will either support the possibility of an anomaly as defined by the experts or not. Assumptions of default values are much more common in abductive systems [29, 8] which would receive as input the presence of an anomaly and then would attempt to posit the most likely set of evidence, in this case sensor readings, which might have caused that anomaly to occur.

We choose here an example of this default dilemma from the PTDS domain. Sensors #327 and #328 measure pressure levels on the turbopump's balance piston. They can report any of the following conditions: Nominal, Level Shift $\pm$, or Spike $\pm$.[21] The difference between these two sensors' readings is itself meaningful: "Delta Level Shift 327–328." This difference can be "Positive," "Negative" or "Zero" where a non-zero value indicates a change in the sensors' relative values and "Zero" represents no such change. Input data from the experts include the following four support conditions governing "Delta Level Shift 327–328" [6, 2]:

> It is probable that [Delta Level Shift 327–328] is [Positive] depending upon ...
>
> PID 327 = Level Shift +
> PID 328 = Level Shift -

[19] The prototype inference engine operates using integer linear programming and a variant of the simplex algorithm [21]. In this model the assumption node always has the extreme value, M—"Big M". As a result, any calculable state of the world which does not use this value will always produce a better solution.

[20] Although termed the exception, the probability value of the default support condition could, in fact, be quite high relative to the its sibling nodes. To the inference engine there is no distinguishing characteristic of a default support node. It is a value like any other available for computation.

[21] See Appendix B for graphical example of these values from the sensor data.

It is probable that [Delta Level Shift 327–328] is [Negative] depending upon ...

> PID 327 = Level Shift -
> PID 328 = Level Shift +

It is probable that [Delta Level Shift 327–328] is [Zero] depending upon ...

> PID 327 = Level Shift -
> PID 328 = Level Shift -

It is probable that [Delta Level Shift 327–328] is [Zero] depending upon ...

> PID 327 = Level Shift +
> PID 328 = Level Shift +

This random variable highlights the difficulties of reasoning with defaults as there exist no explicit support conditions to define Delta Level Shift's value under any other circumstances, *e.g.*, those when either sensor is spiking up or down or both are. It is precisely this situation which occurs as the reasoner attempts to determine the likelihood of "Anomaly 5.06.1".[22] A "Spike" value in either sensor and "Zero" for Delta Level Shift are the prerequisites for this anomaly, but the value of Delta Level Shift cannot be inferred in this example unless both sensors register a level shift.

A "default as superset" assumption that "Delta Level Shift" has value "Zero" breaks the logjam, but also runs afoul the expert's determination that "Delta Level Shift should not be zero" except in certain cases. Using the BKB's "default as negation" approach, the inference engine assumes the necessary value to allow its reasoning to continue.

6 Conclusions

This research develops a viable knowledge acquisition and maintenance tool and its associated methodology which together implement the new BKB knowledge model. This new tool, MACK, guarantees the consistency of the data stored in a BKB's knowledge base as it is both acquired and later maintained. Moreover, this tool has been applied to a real-world domain—NASA's Post-Test Diagnostic System—which supports Space Shuttle main engine analysis.

[22] "Spike seen in sensor 327 or 328 only, with no change in steady state pressures or pressure difference indicates no real rotor motion and possible anomaly in omni seal or sensor itself." [6]

MACK, which is implemented on an explicit object-oriented analytical foundation, contains routines designed automatically and incrementally to confirm the consistency of the knowledge being received from the expert and provide him/her with natural assistance in the transfer of knowledge. Regular incremental checks preserve both probabilistic validity and logical consistency by flagging the inconsistent data points to the expert as they are entered and presumably under his/her current consideration. Such checking guards against expert oversight—*e.g.*, the "Whoops! I forgot to run the consistency checker again!" phenomenon—and helps prevent information overload since there can be at most five adjustments required of the expert immediately after any given run of the consistency checking module.[23]

The tool is able to accept and manipulate time-dependent data which is both common and required not only in the PTDS domain modelled herein, but in many other real-world applications as well. Moreover, this capability will prove crucial to any eventual efforts to operate a BKB inferencing mechanism in real- time or near real-time. In addition, we have determined the BKB's available methods for dealing with the incomplete information which grant it its flexibility as a knowledge representation. "Default as negation" is the preferred mechanism as it preserves all of the rules and data interactions expressly catalogued by the expert and the ability of the forthcoming Explanation & Interpretation Facility to explain the system's results. This work integrates aspects of three disparate reasoning schemes—probabilistic reasoning, temporal reasoning, reasoning with defaults—into the BKB model, particularly as they touch upon knowledge acquisition.

In order to implement the MACK tool properly to guarantee consistency of the knowledge, we had to formalize the notion of consistency for BKBs and then determine the necessary conditions and constraints. The constraints ensure the proper relationships between the instantiation and support nodes are in force at all times. This includes algorithms both to detect constraint violations and to facilitate corrections.[24]

We have availed ourselves of the BKB's computational efficiencies without sacrificing the increased structural flexibility they afford both for reasoning with uncertainty and when compared to other Bayesian inferencing methods. We

[23]The consistency checking routine runs after each addition, removal or edit of *any* support node, following the removal an instantiation node, and after receipt of the fifth consecutive instantiation.

[24]An interesting challenge in the enumeration of these constraints lies in the development of the mathematical specifications from which they were built. For example, Constraint 5's cycles can be of any length ≥ 2 and there are a combinatorial number of support node permutations for these cycles based on Stirling numbers of the First Kind [17]. Meanwhile, the groups of tail-compatible support nodes which must all sum to less than 1 in Constraint 8 are also of indeterminate size. Automating this constraint by itself required separate routines to find and define five distinct sets of three of the different object types that make up a BKB for each sum! After that, normalization was little more than an afterthought. For additional details see [32].

evidence this by the construction of a BKB for the Post-Test Diagnostics System. Continued work in the application domain by PESKI and BKB researchers promises to facilitate on-going PTDS knowledge acquisition within NASA as well as to provide the agency with novel, probabilistic reasoning alternatives.

MACK completes PESKI's Knowledge Acquisition and Maintenance module (see Figure C.1). With that foundation established by this work, follow-on research will now concentrate on the development of the other components of the PESKI architecture, namely the Inference Engine and, just as importantly, the Explanation & Interpretation Facility. Future directions for work with the MACK tool include its application to a new domain, one for which there has been no previous knowledge engineering, and development of a graphical user interface better to interact with the expert. This could include, but is not limited to, "point-and-click" BKB construction and on-screen representations of the BKB's inconsistencies which should facilitate the expert's comprehension of the knowledge model.

With MACK, by allowing the expert to enter his/her data, conduct verification test runs, and receive from those tests an explanation detailed enough to allow the expert to refine and adjust the knowledge base appropriately, this will establish BKBs as a new methodology for reasoning under uncertainty.

References

[1] Space transportation system training data, SSME orientation (part a - engine). Rocketdyne Division, Rockwell International Corporation, 1988.

[2] Reusable rocket engine turbopump health management system. Technical Report Final Report, Contract NAS3-25882, Science Applications International Corporation, 1994.

[3] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[4] Fahiem Bacchus. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities.* The MIT Press, 1990.

[5] Darwyn Banks. Acquiring consistent knowledge for bayesian forests. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology, 1995.

[6] Timothy W. Bickmore. Personal communication. NASA Aerojet Propulsion Division, Sacramento, CA, 1994.

[7] Bruce G. Buchanan and Edward H. Shortliffe. *Rule-Based Expert Systems.* Addison Wesley, 1984.

[8] Eugene Charniak and Solomon E. Shimony. Cost-based abduction and MAP explanation. *Artificial Intelligence*, 66:345–374, 1994.

[9] Didier Dubois, Henri Prade, Lluis Godo, and Ramon Lopez de Mantaras. A symbolic approach to reasoning with linguistic quantifiers. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 74–82, 1992.

[10] Christopher Elsaesser and Max Henrion. Verbal expressions for probability updates: How much more probable is 'much more probable'? In Max Henrion, Ross D. Shachter, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5.* Elsevier Science Publishers, 1990.

[11] E. A. Feigenbaum. *Knowledge Engineering: The Applied Side of Artificial Intelligence.* Edward A. Feigenbaum, 1980.

[12] Brian R. Gaines and Mildred L. G. Shaw. Eliciting knowledge and transferring it effectively to a knowledge-based system. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):4–13, 1993.

[13] Angelo Gilio. Conditional events and subjective probability in management of uncertainty. In Bernadette Bouchon-Meunier et al., editor, *Uncertainty in Intelligent Systems.* Elsevier Science Publishers, 1993.

[14] Moises Goldszmidt and Judea Pearl. Reasoning with qualitative probabilities can be tractable. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 112–120, 1992.

[15] Avelino J. Gonzalez and Douglas D. Dankel. *The Engineering of Knowledge-Based Systems: Theory and Practice.* Prentice-Hall, Inc., 1993.

[16] Robert F. Hink and David L. Woods. How humans process uncertain knowledge: An introduction for knowlege engineers. *AI Magazine*, pages 41–53, Fall 1987.

[17] Bradley W. Jackson and Dmitri Thoro. *Applied Combinatorics with Problem Solving.* Addison-Wesley, 1990.

[18] H. R. Keshavan, J. Barnett, D. Geiger, and Thomas Verma. Introduction to the special section on probabilistic reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:193–195, March 1993.

[19] James Martin and Steven Oxman. *Building Expert Systems: A Tutorial.* Prentice-Hall, 1988.

[20] R. E. Neapolitan. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* John Wiley & Sons, 1990.

[21] George L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors. *Optimization: Handbooks in Operations Research and Management Science Volume 1*, volume 1. North Holland, 1989.

[22] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[23] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[24] Peter G. Raeth. *Expert Systems: A Software Methodology for Modern Applications.* IEEE Computer Society Press, 1990.

[25] Thomas F. Reid. Maintenance in probabilistic knowledge-based systems. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology, 1987.

[26] Kristian Sandahl. Transferring knowledge from active expert to end-user environment. *Knowledge Acquisition*, 6(1):1–22, 1994.

[27] Eugene Santos, Jr. Computing with bayesian multi-networks. Technical Report AFIT/EN/TR93-10, Department of Electrical and Computer Engineering, Air Force Institute of Technology, 1993.

[28] Eugene Santos, Jr. A fast hill-climbing approach without an energy function for finding mpe. In *Proceedings of the 5th IEEE International Conference on Tools with Artificial Intelligence*, 1993.

[29] Eugene Santos, Jr. A linear constraint satisfaction approach to cost-based abduction. *Artificial Intelligence*, 65(1):1–28, 1994.

[30] Eugene Santos, Jr. Unifying time and uncertainty for diagnosis. to appear in the Journal of Experimental and Theoretical Artificial Intelligence., 1994.

[31] Eugene Santos, Jr. and Eric Paul Baenen. On optimal reasoning over large uncertain knowledge bases. in preparation, 1995.

[32] Eugene Santos, Jr. and Darwyn O. Banks. A probabilistic framework for representing uncertainty. in preparation, 1995.

[33] Ross D. Shachter and David E. Heckerman. Thinking backward for knowledge acquisition. *AI Magazine*, pages 55–61, Fall 1987.

[34] Glenn Shafer. Probability judgement in artificial intelligence and expert systems. *Statistical Science*, 2:3–44, 1987.

[35] David J. Spiegelhalter, Rodney C. G. Franklin, and Kate Bull. Assessment, criticism and improvement of imprecise subjective probabilities for a medical expert system. In Max Henrion, Ross D. Shachter, L.N. Kanal,

and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5.* Elsevier Science Publishers, 1990.

[36] D. A. Waterman. *A Guide to Expert Systems.* Addison-Wesley, 1986.

A Selected Excerpts from Interviews for HPOTP

This appendix juxtaposes the output text from Aerojet Propulsion Division's HPOTP expert system and the BKB support node(s) developed in this work with the actual input from the expert [6]. For each of the five anomalies highlighted herein, we identify first the expertise, followed by the Aerojet anomaly(-ies) and BKB support conditions in that order.

Anomaly 5.06.1:

- Expert Input from knowledge acquisition interview transcripts:
  "Seeing change in one, [but] not the other probably is due to static seal in housing or pressure shift not associated with real rotor motion. It probably is not a sensor problem."
- Anomaly Report Text:
  "Spike seen in sensor <327—328> only, with no change in steady state pressures or pressure difference. Possible sensor or omni seal anomaly. No real rotor motion."
- BKB Support Conditions:
  - It is probable that the [Anomaly 5.06.1] is [Found] depending upon ...

    PID 327 = Spike -
    PID 328 = Spike 0
    Delta Level Shift for PIDs 327 & 328 = Zero

  - It is probable that the [Anomaly 5.06.1] is [Found] depending upon ...

    PID 327 = Spike +
    PID 328 = Spike 0
    Delta Level Shift for PIDs 327 & 328 = Zero

  - It is probable that the [Anomaly 5.06.1] is [Found] depending upon ...

    PID 328 = Spike -
    PID 327 = Spike 0
    Delta Level Shift for PIDs 327 & 328 = Zero

  - It is probable that the [Anomaly 5.06.1] is [Found] depending upon ...

    PID 328 = Spike +
    PID 327 = Spike 0

Delta Level Shift for PIDs 327 & 328 = Zero

Anomaly 5.06.2:

- Expert Input from knowledge acquisition interview transcripts:
  "Level change in one and not in the other .... I don't believe we can see cup seal/washer failures or problems. Changes were made to the design to eliminate this type of problem. Also it is highly unlikely that a piece (of seal) could migrate to a pressure opening an effect that pressure. I would be skeptical that it is a cup washer, more likely it is an omni seal or a sensor problem."
- Anomaly Report Text:
  "Level shift seen in sensor <327—328> only. Possible sensor problem, omni seal leakage problem. No real rotor motion."
- BKB Support Conditions:
  - It is possible that the [Anomaly 5.06.2] is [Found] depending upon ...

    PID 327 = Level Shift -
    PID 328 = Level Shift 0

  - It is possible that the [Anomaly 5.06.2] is [Found] depending upon ...

    PID 327 = Level Shift +
    PID 328 = Level Shift 0

  - It is possible that the [Anomaly 5.06.2] is [Found] depending upon ...

    PID 328 = Level Shift -
    PID 327 = Level Shift 0

  - It is possible that the [Anomaly 5.06.2] is [Found] depending upon ...

    PID 328 = Level Shift +
    PID 327 = Level Shift 0

Anomaly 5.06.4:

- Expert Input from knowledge acquisition interview transcripts:

"See the delta is 327–328 so if this one goes up and this one goes down than this should go up and its possibly anomalous rotor motion considering this is constant power level."

- Anomaly Report Text:
  "Possible HPOTP anomalous rotor motion."
- BKB Support Conditions:
  - It is possible that the [Anomaly 5.06.4] is [Found] depending upon . . .

    PID 327 = Level Shift -
    PID 328 = Level Shift +
    Delta Level Shift for PIDs 327 & 328 = Negative

  - It is possible that the [Anomaly 5.06.4] is [Found] depending upon . . .

    PID 327 = Level Shift +
    PID 328 = Level Shift -
    Delta Level Shift for PIDs 327 & 328 = Positive

Anomalies 5.15.1 & 5.15.2:

- Expert Input from knowledge acquisition interview transcripts:
  "951 is one of the [sensors] that addresses the pressure in the LOX [liquid oxygen] drain. 1187 is the temperature that I showed you that you can pick whether it's a new pump or not. 95% of the time it runs at 160 [psia] upwards to 450. I think from an erratic criteria needs to be able to discriminate. The thing that bothers me here is that 951 could be erratic for cause and it would not necessarily cause 1187 temperature measurement to be erratic. So should not try to couple the two measurements. The thing to do would be to classify either or both as being erratic and go from there. I believe that I would do 951 like we did the other pressures. Establish the data base and you compare each test against that database. You also analyze for erratic behavior and if it by itself falls out than flag it. The temperature of the erratic test is appropriate. There are two characteristics and one or the other is always there. What you look for is something different."
- Anomaly Report Text:
  "HPOTP erratic primary pump seal drain pressure may indicate sensor problem or seal anomaly. No effect seen in drain temperature."

- BKB Support Conditions:
  - It is possible that the [Anomaly 5.15.1] is [Found] depending upon ...

    PID 951/952/953 = Erratic
    PID 1187 = Nominal

  - It is possible that the [Anomaly 5.15.1] is [Found] depending upon ...

    PID 951/952/953 = Spike
    PID 1187 = Nominal

  - It is possible that the [Anomaly 5.15.2] is [Found] depending upon ...

    PID 1187 = Erratic
    PID 951/952/953 = Nominal

  - It is possible that the [Anomaly 5.15.2] is [Found] depending upon ...

    PID 1187 = Spike
    PID 951/952/953 = Nominal

B Temporal Parsing

This appendix outlines the methodology by which a test run is parsed into the time slices discussed in Section 5.1. At present, MACK uses a similar method, but this capability is neither automated nor a part of the tool itself. Instead, we simply pre-process the temporal data by hand before entering it.

In Figure B.1 we have constructed a very simplistic example. The figure contains data streams from three hypothetical sensors over a ten-second period divided into one-second intervals. The algorithm assigns breakpoints in the timeline at the beginning and end of any event *e.g.*, spikes, peaks, level shifts from any of the sensors. The result is a series of intervals as shown.

Periods 1, 4, 6, 8 and 11 contain no significant events, *i.e.*, all sensors are nominal. Positive spikes/peaks occur in periods 2 and 3; negative spikes in 7, 9 and 10. We see level shifts during periods 5, 7 and 12, and erratic readings from PID 1 in the last interval. Notice that period 7, originally subdivided into three by the spike in PID 2, is unified to reflect the occurrence of a single level shift in PID 3 rather than three distinct shifts which would be recognizable by two small plateaus during the decline.